APPENDIX

The fitting of the classifiers in this study can be implemented in Matlab via the Statistics Toolbox. Here we are the commands needed for generating the trained classifier:

- logistic regression: fitglm(X, Y, 'Distribution', 'binomial', 'link', 'logit');
- linear discriminant: fitcdiscr(X, Y, 'DiscrimType', 'linear', 'Gamma', 0, 'FillCoeffs', 'off', 'ClassNames', [0; 1]);
- Naive Bayes: fitcnb(X, Y, 'ClassNames', [0; 1]);
- complex tree: fitctree(X, Y, 'SplitCriterion', 'gdi', 'MaxNumSplits',100, 'Surrogate', 'off', 'ClassNames', [0; 1]);
- medium tree: fitctree(X, Y, 'SplitCriterion', 'gdi', 'MaxNumSplits', 20, 'Surrogate', 'off', 'ClassNames', [0; 1]);
- simple tree: fitctree(X, Y, 'SplitCriterion', 'gdi', 'MaxNumSplits', 4, 'Surrogate', 'off', 'ClassNames', [0; 1]);
- linear SVM: fitcsvm(X, Y, 'KernelFunction', 'linear', 'PolynomialOrder', [], 'KernelScale', 'auto', 'BoxConstraint', 1, 'Standardize', true, 'ClassNames', [0; 1]);
- quadratic SVM: fitcsvm(X, Y, 'KernelFunction', 'polynomial', 'PolynomialOrder', 2, 'KernelScale', 'auto', 'BoxConstraint', 1, 'Standardize', true, 'ClassNames', [0; 1]);
- cubic SVM: fitcsvm(X, Y, 'KernelFunction', 'polynomial', 'PolynomialOrder', 3, 'KernelScale', 'auto', 'BoxConstraint', 1, 'Standardize', true, 'ClassNames', [0; 1]);
- fine KNN: fitcknn(X,Y, 'Distance', 'Euclidean', 'Exponent', [], 'NumNeighbors', 1, 'DistanceWeight', 'Equal', 'Standardize', true, 'ClassNames', [0; 1]);
- medium KNN: fitcknn(X,Y, 'Distance', 'Euclidean', 'Exponent', [], 'NumNeighbors', 10, 'DistanceWeight', 'Equal', 'Standardize', true, 'ClassNames', [0; 1]);
- cosine KNN: fitcknn(X,Y, 'Distance', 'Cosine', 'Exponent', [], 'NumNeighbors', 10, 'DistanceWeight', 'Equal', 'Standardize', true, 'ClassNames', [0; 1]);
- cubic KNN: fitcknn(X,Y, 'Distance', 'Minkowski', 'Exponent', 3, 'NumNeighbors', 10, 'DistanceWeight', 'Equal', 'Standardize', true, 'ClassNames', [0; 1]);
- weighted KNN: fitcknn(X,Y, 'Distance', 'Euclidean', 'Exponent', [], 'NumNeighbors', 10, 'DistanceWeight', 'SquaredInverse', 'Standardize', true, 'ClassNames', [0; 1]);
- ensemble: bagged trees: fitcensemble(X, Y, 'Method', 'Bag', 'NumLearningCycles', 30, 'Learners', templateTree('MaxNumSplits', 91), 'ClassNames', [0; 1]);
- ensemble: subspace discriminant: fitcensemble(X, Y, 'Method', 'Subspace', 'NumLearningCycles', 30, 'Learners', 'discriminant', 'NPredToSample', max(1, min(29, size(X,2) - 1)), 'ClassNames', [0; 1]);
- ensemble: Ssubspace KNN: fitcensemble(X, Y, 'Method', 'Subspace', 'NumLearningCycles', 30, 'Learners', 'knn', 'NPredToSample', max(1, min(29, size(predictors,2) - 1)), 'ClassNames', [0; 1]);
- ensemble: RUSboosted trees: fitcensemble(X, Y, 'Method', 'RUSBoost', 'NumLearningCycles', 30, 'Learners', templateTree('MaxNumSplits', 20), 'LearnRate', 0.1, 'ClassNames', [0; 1]);

The L1-regularization of lasso for linear regression (lasso: lassoglm(X, Y,'binomial','CV',15)) uses a 15-fold CV to select the lambda which provides the minimum deviance, assuming a binomial distribution of the response. This is run inside a 27-fold CV on the original dataset.